

# Introduction to Programming

with Java, for Beginners

Primitive vs. References Type  
The Stack and the Heap

## Primitive vs. Reference Types

- We've seen Java's 4 *primitive* types:  
int, double, boolean, char
- Types other than the primitive types are known as *reference* types
  - Used for objects
- Examples of reference variables:  
Dog d1; Counter c1; Player mario; String name;  
Note: String is an object not primitive type

ESE112

2

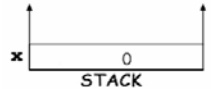

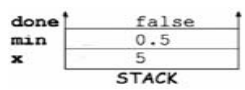
## Memory: Stack and Heap

- When we run Java programs, memory is allocated for variables and objects
- Understanding how this memory is managed helps us understand how Java works
- The JVM uses two kinds of memory: *stack* and *heap*
- The *stack* is used to store variables of primitive type:
  - When created in the DrJava interactions pane
  - During method calls
- The *heap* is used to store *objects*

ESE112

3

## How the Stack Works

DrJava Interactions	Stack
> int x;	 <p>A diagram of a stack with two vertical arrows. A horizontal bar represents a stack element. The bar is labeled 'x' on the left and '0' in the center. Below the bar is the word 'STACK'.</p>
> x = 5;	 <p>A diagram of a stack with two vertical arrows. A horizontal bar represents a stack element. The bar is labeled 'x' on the left and '5' in the center. Below the bar is the word 'STACK'.</p>
> double min = 0.5; > boolean done = false;	 <p>A diagram of a stack with two vertical arrows. Three horizontal bars represent stack elements. The top bar is labeled 'done' on the left and 'false' in the center. The middle bar is labeled 'min' on the left and '0.5' in the center. The bottom bar is labeled 'x' on the left and '5' in the center. Below the bars is the word 'STACK'.</p>

Note: Variables are added in the order they are declared

ESE112

4

## Reference Type

- In Java, no variable can ever hold an entire object
  - One variable can only contain one thing
  - Object consists of multiple of data/state and hence stored on *heap*
- The term *reference* is used because it *refers* to a memory location where the object lives
  - The variable of reference type is used to access the object
- The value of reference variable is either “null” or a “heap address”
  - *null* means currently not pointing at any location

ESE112

5

## Value of a Reference Variable

Example:

```
> Counter c1;
> c1
null
> c1 = new Counter();
> c1
Counter@e05ad6
```

- e05ad6 is location in memory where c1 resides
  - e05ad6 hexadecimal (base 16) number
  - This location will differ on your computer
- We don't have to (and can't) deal with these hex numbers directly
  - Convenience of using variables

ESE112

6

## How the Heap Works

DrJava Interactions	Stack and Heap
<pre>&gt; int x = 99; &gt; Counter c1; &gt; c1 null</pre>	<p>The diagram shows a vertical line representing the stack. On the left, two boxes are shown: the top one is labeled 'c1' and contains 'null'; the bottom one is labeled 'x' and contains '99'. To the right of the stack is a horizontal line representing the heap, which is currently empty.</p>
<pre>&gt; c1 = new Counter(); &gt; c1 Counter@2f996f</pre>	<p>The diagram shows the stack with 'c1' containing 'Counter@2f996f' and 'x' containing '99'. On the heap, a circle labeled '0' is shown with an arrow pointing from the 'c1' box in the stack to it.</p>
<pre>&gt; c1.incrementCount(); &gt; Counter c2 = new Counter(); &gt; c2 Counter@4a0ac5</pre>	<p>The diagram shows the stack with 'c2' and 'c1' both containing 'Counter@4a0ac5', and 'x' containing '99'. On the heap, two circles labeled '0' and '1' are shown. Arrows point from both the 'c1' and 'c2' boxes in the stack to the '0' object on the heap.</p>

ESE112

7

## Aliases

- Two or more references can point to the same object
  - These references are then known as aliases
- Example (In Dr Java Interactions Pane)
 

```
> Dog d1 = new Dog("Lassie", 5);
> d1
Dog@83d8be
> Dog d2 = d1;
> d2
Dog@83d8be
> d1.getAge()
5
> d2.getAge()
5
```

ESE112

8

## String

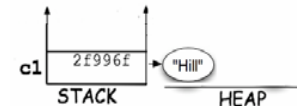
- A sequence of characters
- A String is a built-in Java object type
- Java provides this type because it's used so frequently
- Examples of String creation:

```
> String s1 = new String("hello");
> String s2 = "hello"; // commonly used shortcut
> s2 + " you!"
"hello you!"
> s2 = "The result is " + 100;
> System.out.println(s2);
"The result is 100"
```

ESE112

9

## String (contd..)

DrJava Interactions	Stack and Heap
<pre>&gt; String c1 = new String("Hill");</pre>	

```
> String c1 = new String ("Hill");
> c1
"Hill"
```

Note:

- We do not get heap address of String reference
- Later, when we learn "Inheritance" it will be clear

ESE112

10